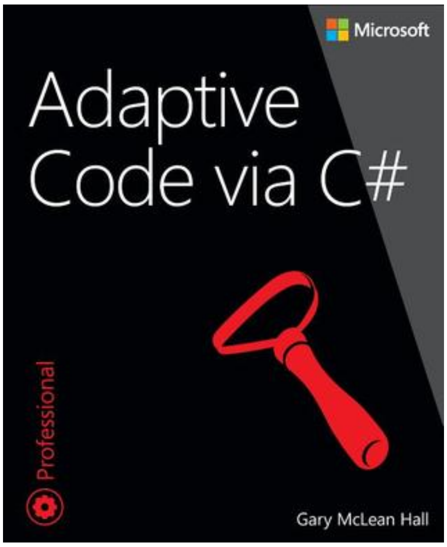

Adaptive Code Via C#: Agile Coding with Design Patterns and Solid Principles

Gary McLean Hall

Adaptive Code Via C#: Agile Coding with Design Patterns and Solid Principles

Gary McLean Hall

Adaptive Code Via C#: Agile Coding with Design Patterns and Solid Principles Gary McLean Hall Your process may be agile, but are you building agility directly into the code base? This book teaches .NET programmers how to give code the flexibility to adapt to changing requirements and customer demands by applying cutting-edge techniques, including SOLID principles, design patterns, and other industry best practices.

Understand why composition is preferable to inheritance and how flexible the interface really can be Gain deep knowledge of key design patterns and anti-patterns, when to apply them, and how to give their code agility Bridge the gap between the theory behind SOLID principles, design patterns, and industry best practices by pragmatically solving real-world problems Get code samples written in upcoming version of Microsoft Visual C# Topics include: Agile with Scrum process; dependencies and layering; the interface; patterns and anti-patterns; introduction to SOLID principles, including open/closed and dependency interjection; and using application templates

Adaptive Code Via C#: Agile Coding with Design Patterns and Solid Principles Details

Date	:	Published May 22nd 2014 by Microsoft Press (first published February 22nd 2014)
ISBN	:	9780735683204
Author	:	Gary McLean Hall
Format	:	Paperback 544 pages
Genre	:	Computer Science, Programming, Technical, Science, Technology, Software, Computers

Download and Read Free Online Adaptive Code Via C#: Agile Coding with Design Patterns and Solid Principles Gary McLean Hall

From Reader Review Adaptive Code Via C#: Agile Coding with Design Patterns and Solid Principles for online ebook

Mark Magagna says

Definitely worth reading.

Pandiyan says

Good one :)

Artem Valieiev says

I suggest this book everyone who are going to work in team. It's covers basics of scrum process and management, with fundamentals of design practices in C# and solid Principles.

Jose Seco Sanz says

Great book.

Ben Rand says

A really concise overview of several important programming concepts. Many of the main topics here (Dependency Injection, Patterns) have entire books dedicated to those topics so this is either a good starting point or good review. The best thing about the book, IMO, is how it glues together a lot of these concepts.

Arun Mahendrakar says

This book is a great read for C# developers. Depending on your level of experience, you may find it either highly informative if you're a mid-level developer or use it as a great reference book if you're an experienced developer.

The book starts with Scrum basics. It was reviving to read about the overall process. There are some crucial design strategies that are described in part 1 of the book.

The chapter about Unit testing and refactoring in part 1 was very convincing as well. The explanation of TDD was quite detailed and in an easy-to-understand language.

Part II is dedicated to SOLID principles and their implementation. Each of these chapters are as in-depth as one can get. I have not read a better explanation of Liskow substitution principle anywhere else. I finally understand that principle.

Part III narrates a team discussion on how SCRUM and SOLID principles come together. It was done in a very 'story'-ish way so it was nice to read through that conversation.

There's an introduction about Git in the Appendix and if you're new to Git this will be a nice overview to get started with.

Edonis Seferi says

No book made clear the solid principles as well as this. I can not recommend this highly enough. I am seriously considering reading the new revision that has recently been published. It is extremely concise, not verbose at all (unlike some software development books that attempt to be far too intelligent diluting the intended message) and the examples are brilliant.

Any developer can learn from this, from beginner to senior. Simplicity is the ultimate sophistication.

Tom Jones says

Highly recommended.

I've been using C# for a while (My first Programming language) and learned new things and concepts about the language.

For someone who's looking into a career as a software developer primarily using C#, it was very helpful. Useful for both beginners who wish to learn the language and someone like myself.

5 stars from me.

Pawel says

It's the best programming book I've read for a long time. When I was reading it there were so many new and fascinating things that I felt like Neo in Matrix learning whole Kung Fu in just a few minutes.

Just a few examples of the things I learned (and there were many, many others):
-I already knew to favor composition over inheritance but I learned why
-why using "new" keyword is (in many cases) bad
-how to create more readable/declarative tests using the builder pattern
-2 separate types of TDD: purist and pragmatic
-to focus more on MTTR (Mean Time To Recovery) than on MTBF (Mean Time Before Failure)
-I gained a much deeper understanding of SOLID principles
-what is the purpose of exceptions ;)

I recommend this book to any developer:
-for junior devs it will be very dense with knowledge and may require to be read a few times
-for mid-level devs (like me) it will be very rewarding
-for senior devs it may be a great refresher and a summary of the knowledge

---

Jeff Patterson says

This book changed the way I think about programming.

Daniel Oliver says

A good primer for a typical C# developer in Nashville.

Aelena says

Excellent book. Wish I had found a book like this 10 years earlier. Even seasoned devs or architects will find here concepts and ideas they did not know before or hadn't understand with the necessary clarity. That is precisely one of the virtues of the book. Simple, clear, concise and precise language, with no unwarranted jargon o verbosity. Make sure you get a copy for yourself and a few copies for your younger and more eager devs.

Charles Mccown says

This is an excellent resource for learning about the SOLID principles.

The book begins with an overview of Scrum, unit testing and refactoring.

Next it gives excellent explanations for each of the principles. I was very happy with these chapters which were well formatted, easy to understand and a joy to read.

Finally the book concludes with samples that follow a team of developers through the agile process. These felt genuine, and helped reinforce the material.

Great book for developers interested in making their future selves lives easier.

Dmitriy Melnik says

The book covers a big deal of best principles and practices of OOP development. As an experienced .NET developer, I found a few things new to me, though most of the material I already knew. As a whole the book was of a little value to me. I think it is most valuable for junior developers.

Dan Dexter says

What caught my attention about Adaptive Code Via C# is that it combined Agile methodologies, design patterns, and SOLID Principles into one book and this was made readily apparent by the subtitle. When I came across this book, I was more interested in the SOLID Principles part as I've had a decent amount of experience working on Agile teams, and had a copy of the well-received Design Patterns book by the Gang Of Four, which I was certain that I wouldn't learn anything new from Adaptive Code Via C# and that it would just be the same material. I was wrong. What I found while reading this book is that the author describes the Agile methodology in a way that is refreshing even if you've been on Agile teams before. I was also pleasantly surprised at how well the author described some design patterns in a way that wasn't overly technical, but instead in a way that is more conversational and in easily digestible chunks.

The book does a good job of marrying the topics of Agile, design patterns, and SOLID together by describing the relationship between them and how they complement one another. The author starts off with an overview of Agile, the need for adaptive code in order to adapt to rapidly changing requirements, and the importance of managing dependencies and creating a layered architecture such that the overall architecture is structured in a way that promotes adaptability. As part of the same section, he also talks about design patterns and several in particular that lend themselves nicely to adaptive code. From there, the author then describes the SOLID Principles, what they are, what they mean, and how following the principles helps create code that is easily extendable and easily unit tested. I was aware of the SOLID Principles in theory, but this book included a lot of example code both before and after each principle was applied that really drove home just how valuable these principles can be when applied correctly. Lastly, the book finishes with a simulated Agile project that steps you through a couple of simulated sprints on a project, the dialogue the developers have, and how the design patterns and SOLID Principles are applied in a way that makes future additions much easier to integrate into the system.

Overall this book was very well written, fun to read, and I highly recommend it. If you're not experienced with C# much, I wouldn't let that prevent you from getting the book, a lot of the knowledge is easily applied to other languages. The author states that the target audience for this book is intermediate level developers, which I don't necessarily disagree with, but I could see junior engineers learning a lot from this book provided that they are at a point that they at least have cursory understanding of design patterns and separation of logic. The piece on Agile is also a great refresher for those who have experience with it, but I imagine it is a valuable read for those who are new to the methodology.